MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

SUPPLEMENTARY EXAMINATION

TRIMESTER 1, 2015/2016

ECE3136 – JAVA TECHNOLOGY

(All Sections / Groups)

17 NOV 2015
9.00 AM – 11.00 AM
(2 HOURS)

INSTRUCTIONS TO STUDENT

1. This examination paper consists of SIX (6) pages including cover with THREE (3) questions only.

2. Answer ALL questions. All questions carry 25 marks and the distribution of the marks for each question is given.

3. This is an Open-Book examination. Materials permitted are textbooks and bound notes.

4. Please print all your answers in the Answer Booklet provided.

Question 1

This question is to create classes to facilitate student information records. Your program should contain the following classes:

(a) `Student`

(i) Create a class `Student` which has four instance variables: `name` (type `String`), `year` (type `int`), `faculty` (type `String`) and `id` (type `String`). Provide accessor methods for all variables, and a `toString` method.

[6 marks]

(ii) Provide a constructor. The constructor takes in values for the student's name, year and faculty and sets them using their respective mutators. A call is then made to mutator `setID` to set the student's ID according to a pattern (see part (iii)).

[2 marks]

(iii) Provide mutator methods as follows:

1. The mutator for `name` should set the name. The mutator for `year` should check that the year is a year from 1997 to 2015 before allowing it to be set.

[2 marks]

2. The mutator for `faculty` should check that the value is either "IT", "management" or "engineering" (ignoring case) before allowing it to be set, else create and throw an `InvalidFacultyException` object (see part (b)).

[5 marks]

3. The mutator for `id` sets `id` according to this pattern: the first character of `id` is the first letter of the faculty, followed by the last two digits of the year, followed by 5 random integers.

[4 marks]

(b) `InvalidFacultyException`

As part of the exception handling, create an `Exception` class, `InvalidFacultyException`, which prints out an error message whenever the faculty entered by the user is invalid.

[2 marks]

(c) Create a simple driver class to test `Student` class. The driver program must include erroneous arguments to test the effectiveness of the exception handling.

[4 marks]

Continued...

Question 2

Create a class **`LineGraph`** which draws a line connecting two end points based on input given by the user.

Your program should contain the following:

(a) The class **`LineGraph`** which inherits from **`JPanel`** and has four labels, four text fields and a button. It also has 4 instance variables, which are constants to represent the actual starting and ending of the x-axis and y-axis displayed in Figure Q2. The size of the graph is 400 by 400 pixels. The origin (0,0) shown in Figure Q2 is actually at (50,500) of the Java coordinate system. The arrangement is as shown in Figure Q2.

[6 marks]

(b) Provide a constructor. Instantiate the coordinate text fields as follows: x-coordinate start point = 10, y-coordinate start point=0, x-coordinate start point = 400, y-coordinate start point=400.

[7 marks]

(c) Override **`paintComponent`** method: Draw the x and y axes as shown. Draw the initial line with the default coordinates in part (b) and display the coordinates as strings in the graph.

[6 marks]

(d) Refresh the display every time the button is pressed. That is, draw the new line using the user entered values, calculate the gradient and update the strings displayed in the graph.

[6 marks]

Note: DO NOT write the driver program.

Continued...

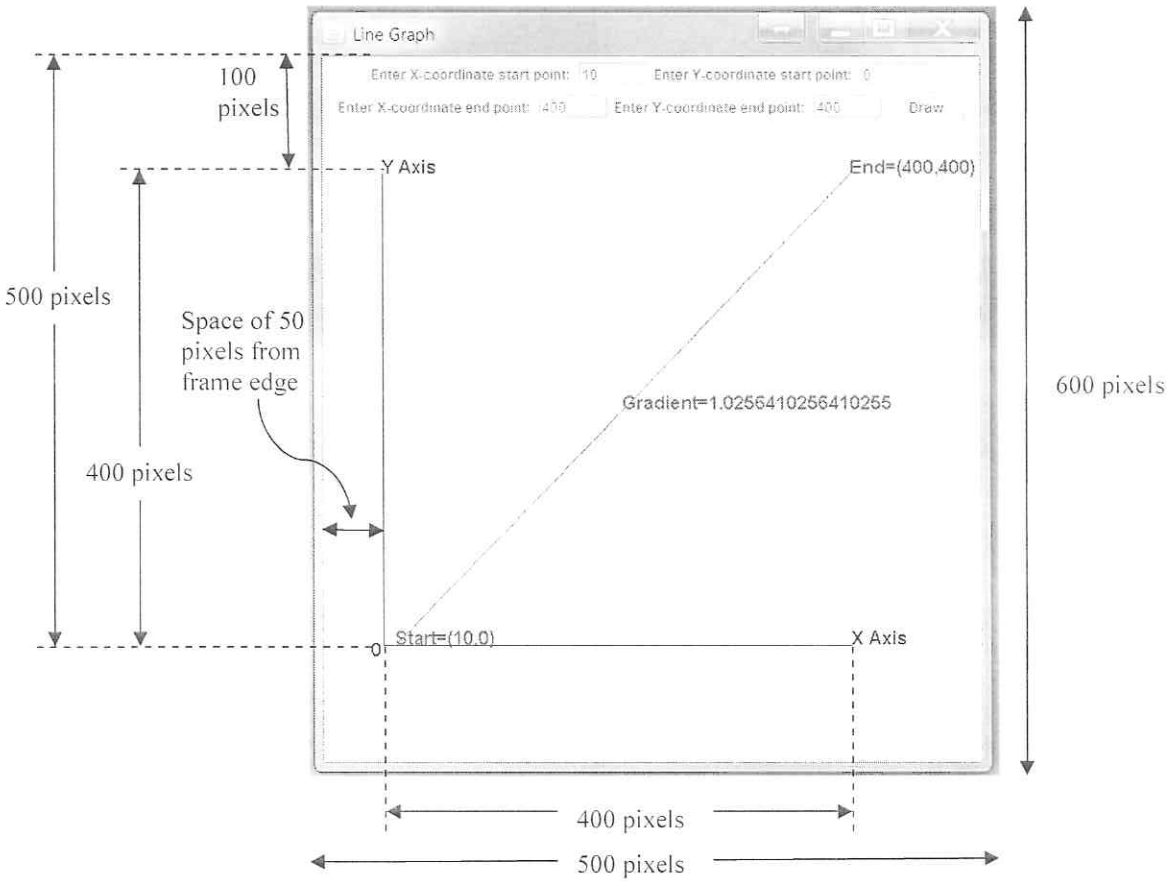

Figure Q2

Continued...

Question 3

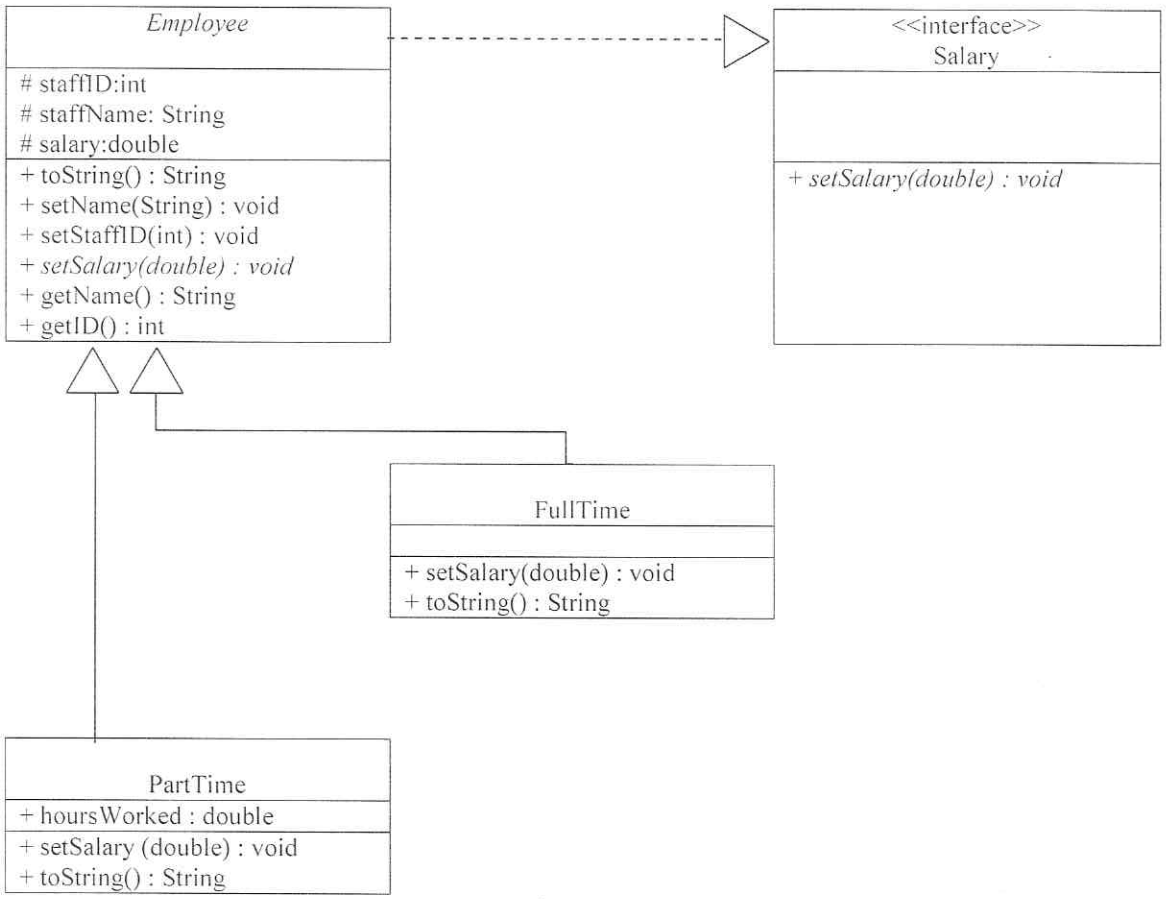

Figure Q3

Refer to the UML diagram in Figure Q3. In your programs, include all class members shown in the UML diagram.

(a) Create abstract class **`Employee`** which implements interface **`Salary`**.

[2 marks]

(i) Provide mutator and accessor methods for **`staffID`** and **`staffName`**. The mutator must only allow the value of **`staffID`** to be set to a number ranging from 10000 to 99999. Otherwise, output an error message and set **`staffID`** to 0.

[5 marks]

(ii) Provide a constructor that sets **`staffID`** and **`staffName`** using the respective mutators.

[2 marks]

(iii) Provide a **`toString`** method.

[1 mark]

(iv) Provide abstract method **`setSalary`**.

[1 mark]

Continued...

(b) Create interface `Salary`.

[2 marks]

(c) Create a class `FullTime` which inherits from `Employee`.

(i) Provide a constructor that takes in the `staffName`, `staffID` and `salary` and sets the values using the superclass constructor and method `setSalary`. Provide a `toString` method that calls the superclass `toString` method.

[4 marks]

(ii) Override abstract method `setSalary`. The method `setSalary` should only allow the `salary` to be set to a positive value. Output an error message and set `Salary` to 0 otherwise.

[2 marks]

(d) Create a class `PartTime` which inherits from `Employee`. It has an additional instance variable `hoursWorked` which indicates the hours worked by the Part Time employee.

(i) Provide a constructor that takes in the `staffName`, `staffID` and `hoursWorked` and sets `staffName`, `staffID` and `salary` values by calling the superclass constructor and method `setSalary`. Provide a `toString` method that calls the superclass `toString` method.

[4 marks]

(ii) Override abstract method `setSalary`. The method `setSalary` should calculate the `salary` by this formula:

Salary = hoursWorked*6.5

Hours worked must be between 1 and 100. Otherwise, output an error message and set `Salary` to 0.

[2 marks]

Note: DO NOT write the driver program.

End of Paper